DOCKET NO.: 2013P125

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re the Application of:

BONG WAN KIM, ET AL.

Application No.:

Filed:

For: **Apparatus and Method for Controlling Memory Allocation for Vairable Size Packets**

Art Group:

Examiner:

Commissioner for Patents
P.O, Box 1450
Alexandria, VA 22313-1450

REQUEST FOR PRIORITY

Sir:

Applicant respectfully requests a convention priority for the above-captioned application, namely:

COUNTRY	APPLICATION NUMBER	DATE OF FILING
Korea	2002-0081383	18 December 2002

☒ A certified copy of the document is being submitted herewith.

Respectfully submitted,

Blakely, Sokoloff, Taylor & Zafman LLP

Dated: 11/14/03

Eric S. Hyman, Reg. No. 30,139

12400 Wilshire Boulevard, 7th Floor
Los Angeles, CA 90025

KOREAN INTELLECTUAL PROPERTY OFFICE

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

Application Number:: Korean Patent Application 2002-0081383

Date of Application:: 18 December 2002

Applicant(s): : Electronics and Telecommunications Research Institute

30 December 2002

COMMISSIONER

[Bibliography]

[Document Name] Patent Application
[Classification] Patent
[Receiver] Commissioner
[Reference No.] 0022
[Filing Date] 18 December 2002
[IPC] G06F

[Title] Apparatus and method for controlling memory allocation for variable sized packets

[Applicant]
[Name] Electronics and Telecommunications Research Institute
[Applicant code] 3-1998-007763-8

[Attorney]
[Name] Youngpil Lee
[Attorney code] 9-1998-000334-6
[General Power of Attorney Registration No.] 2001-038378-6

[Attorney]
[Name] Haeyoung Lee
[Attorney code] 9-1999-000227-4
[General Power of Attorney Registration No.] 2001-038396-8

[Inventor]
[Name] KIM, Bong Wan
[Resident Registration No.] 691005-1932312
[Zip Code] 138-221
[Address] 119-403 Jugong Apt., Jamsil-dong, Songpa-gu Seoul, Rep. of Korea
[Nationality] Republic of Korea

[Inventor]
[Name] KWAK, Dong Yong
[Resident Registration No.] 590806-1222613
[Zip Code] 305-755
[Address] 123-402 Hanbit Apt., Eoeun-dong, Yusong-gu, Daejeon-city Rep. of Korea
[Nationality] Republic of Korea

[Request for Examination]	Requested	
[Purpose]	We file as above according to Art. 42 of the Patent Law, request the examination as above according to Art. 60 of the Patent Law.	
	Attorney	Youngpil Lee
	Attorney	Haeyoung Lee
[Fee]		
[Basic page]	20 Sheet(s)	29,000 won
[Additional page]	11 Sheet(S)	11,000 won
[Priority claiming fee]	0 Case(S)	0 won
[Examination fee]	20 Claim(s)	749,000 won
[Total]	789,000 won	
[Reason for Reduction]	Government Invented Research Institution	
[Fee after Reduction]	394,500 won	
[Transfer of Technology]	Allowable	
[Licensing]	Allowable	
[Technology Training]	Allowable	

[Enclosures]

1. Abstract and Specification (and Drawings) 1 copy

대 한 민 국 특 허 청
KOREAN INTELLECTUAL PROPERTY OFFICE

별첨 사본은 아래 출원의 원본과 동일함을 증명함.

This is to certify that the following application annexed hereto is a true copy from the records of the Korean Intellectual Property Office.

출 원 번 호 : 10-2002-0081383
Application Number PATENT-2002-0081383

출 원 년 월 일 : 2002년 12월 18일
Date of Application DEC 18, 2002

출 원 인 : 한국전자통신연구원
Applicant(s) Electronics and Telecommunications Research Institu

2002 년 12 월 30 일

특 허 청
COMMISSIONER

【서지사항】

【서류명】 특허출원서
【권리구분】 특허
【수신처】 특허청장
【참조번호】 0022
【제출일자】 2002.12.18
【국제특허분류】 G06F
【발명의 명칭】 가변 길이의 패킷 저장을 위한 메모리 관리 장치 및 방법
【발명의 영문명칭】 Apparatus and method for controlling memory allocation for variable sized packets
【출원인】
【명칭】 한국전자통신연구원
【출원인코드】 3-1998-007763-8
【대리인】
【성명】 이영필
【대리인코드】 9-1998-000334-6
【포괄위임등록번호】 2001-038378-6
【대리인】
【성명】 이해영
【대리인코드】 9-1999-000227-4
【포괄위임등록번호】 2001-038396-8
【발명자】
【성명의 국문표기】 김봉완
【성명의 영문표기】 KIM,Bong Wan
【주민등록번호】 691005-1932312
【우편번호】 138-221
【주소】 서울특별시 송파구 잠실동 주공아파트 119-403
【국적】 KR
【발명자】
【성명의 국문표기】 곽동용
【성명의 영문표기】 KWAK,Dong Yong
【주민등록번호】 590806-1222613

【우편번호】 305-755

【주소】 대전광역시 유성구 어은동 한빛아파트 123-402

【국적】 KR

【심사청구】 청구

【취지】 특허법 제42조의 규정에 의한 출원, 특허법 제60조의 규정에 의한 출원심사 를 청구합니다. 대리인
이영필 (인) 대리인
이해영 (인)

【수수료】

【기본출원료】	20	면	29,000 원
【가산출원료】	11	면	11,000 원
【우선권주장료】	0	건	0 원
【심사청구료】	20	항	749,000 원

【합계】 789,000 원

【감면사유】 정부출연연구기관

【감면후 수수료】 394,500 원

【기술이전】

【기술양도】 희망

【실시권 허여】 희망

【기술지도】 희망

【첨부서류】 1. 요약서·명세서(도면)_1통

【요약서】

【요약】

본 발명은 기억소자(이하 메모리라 칭함)를 활용하는 기기에서의 가변 길이(variable size)의 패킷 저장을 위한 메모리 관리 장치 및 방법에 관한 것이다. 상기 메모리 관리 장치는, 소정의 길이를 가지는 복수 개의 서브 데이터 블록들을 구비한 복수 개의 데이터 블록들을 구비하여, 가변 길이의 기억영역 할당 요구시 상기 서브 데이터 블록 및 상기 데이터 블록 중 어느 하나의 단위로 기억영역이 할당되는 데이터 메모리; 포인터를 이용하여 상기 데이터 메모리의 빈 기억영역을 적어도 하나 이상의 목록으로 관리하는 프리 리스트 메모리; 및 상기 목록의 시작 위치정보 및 끝 위치정보를 저장하기 위한 레지스터를 포함한다.

【대표도】

도 1

【명세서】

【발명의 명칭】

가변 길이의 패킷 저장을 위한 메모리 관리 장치 및 방법{Apparatus and method for controlling memory allocation for variable sized packets}

【도면의 간단한 설명】

도 1은 데이터 메모리 블록 당 서브 블록이 4 개로 구성된, 본 발명의 바람직한 실시예에 따른 메모리 장치의 기억영역의 할당 예를 보여주는 블록도이다.

도 2는 데이터 메모리 블록 당 서브 블록이 2 개로 구성된, 본 발명의 바람직한 실시예에 따른 메모리 장치의 기억영역의 할당 예를 보여주는 블록도이다.

도 3은 도 2에 도시된 메모리 장치를 위한 기억영역 할당 방법을 보여주기 위한 흐름도이다.

도 4는 도 2에 도시된 메모리 장치를 위한 기억영역 반환 방법을 보여주기 위한 흐름도이다.

< 도면의 주요 부분에 대한 부호의 설명 >

1100, 2100 : 프리 리스트 메모리

1200, 2200 : 데이터 메모리

1300, 2300 : 레지스터

【발명의 상세한 설명】

【발명의 목적】

【발명이 속하는 기술분야 및 그 분야의 종래기술】

<9> 본 발명은 전기적인 기억소자(이하, 메모리라 칭함)를 효과적으로 사용하는 방법에 관한 것으로, 특히 메모리를 활용하는 기기에서 가변 길이(variable size)의 패킷 저장을 수행하기 위한 메모리 관리 장치 및 방법에 관한 것이다.

<10> 일반적으로, 컴퓨터나 통신기기에서는 새로운 기억영역(memory space)이 필요할 때, 메모리의 특정 부분에 할당(allocation)이라는 과정을 통해 위치를 확보하고, 이를 포인터(pointer)라 부르는 주소 값(address)을 보관하는 또 다른 기억영역을 통해 관리한다. 이를 위해 대부분의 컴퓨터나 통신기기에서는 메모리를 1 개의 고정된 크기로 나누어 관리하는 방법을 사용하고 있다. 그러나, 이 방법은 가변길이 패킷 중 작은 크기를 갖는 패킷에 대해서도 일률적인 메모리 할당을 수행하기 때문에, 메모리 상에 낭비되는 공간이 많이 발생하게 되는 문제점이 있다.

<11> 이와 같은 문제점을 해결하기 위해, 2000년 7월 11일, Sober에 의해 취득된 U.S. Pat. No. 6,088,777, "MEMORY SYSTEM AND METHOD FOR DYNAMICALLY ALLOCATING A MEMORY DIVIDED INTO PLURAL CLASSES WITH DIFFERENT BLOCK SIZES TO STORE VARIABLE LENGTH MESSAGES" 등에서는 메모리를 미리 고정된 크기로 나누지 않고, 기억영역 할당 요구가 있을 때마다 그 요구량에 맞게 할당해 주는 방식을 제안하고 있다.

<12> 그러나, 이 방법은 메모리의 낭비를 최소화할 수 있는 장점은 있지만, 하드웨어로 구현하기에는 복잡하고, 가비지 콜렉션(garbage collection)이라 불리는 기억영역 정리 작업을 주기적으로 수행해야 하므로, 소프트웨어 상에서만 적용 가능한 문제점이 있다.

【발명이 이루고자 하는 기술적 과제】

<13> 본 발명이 이루고자 하는 기술적 과제는, 메모리 할당에 대한 위치 주소 값 관리를 하드웨어로 구현할 수 있도록 메모리를 복수 개의 고정된 크기로 나누어 관리함으로써, 가변 길이의 메모리 요구에 대해, 메모리의 낭비를 최소화할 수 있는 메모리 관리 장치 및 방법을 제공하는 데 있다.

<14> 본 발명이 이루고자 하는 다른 기술적 과제는, 상기 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록매체를 제공하는데 있다.

【발명의 구성 및 작용】

<15> 상기의 과제를 이루기 위하여 본 발명에 의한 메모리 관리 장치는, 소정의 길이를 가지는 복수 개의 서브 데이터 블록들을 구비한 복수 개의 데이터 블록들을 구비하여, 가변 길이의 기억영역 할당 요구시 상기 서브 데이터 블록 및 상기 데이터 블록 중 어느 하나의 단위로 기억영역이 할당되는 데이터 메모리; 포인터를 이용하여 상기 데이터 메모리의 빈 기억영역을 적어도 하나 이상의 목록으로 관리하는 프리 리스트 메모리; 및 상기 목록의 시작 위치정보 및 끝 위치정보를 저장하기 위한 레지스터를 포함하는 것을 특징으로 한다.

<16> 상기의 과제를 이루기 위하여 본 발명에 의한 메모리 할당 방법은, (a) n이 2의 배수이고 i가 양의 정수일 때, 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 큰 경우, 프

리 리스트 메모리에 의해 관리되는 n 바이트 엔트리 목록 상에 존재하는 유효 엔트리에 n 바이트의 기억영역을 할당하는 단계; 및 (b) 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 같거나 작은 경우, 상기 프리 리스트 메모리에 의해 관리되는 $\frac{n}{2^i}$ 바이트 엔트리 목록 상에 존재하는 유효 엔트리에 $\frac{n}{2^i}$ 바이트의 기억영역을 할당하되, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록에 유효 엔트리가 존재하지 않으면, 상기 n 바이트 엔트리 목록을 분할하여 $\frac{n}{2^i}$ 바이트의 기억영역으로 할당하는 단계를 포함하는 것을 특징으로 한다.

<17> 상기의 과제를 이루기 위하여 본 발명에 의한 메모리 반환 방법은, (a) n이 2의 배수이고 i가 양의 정수일 때, 반환되는 기억영역의 크기가 $\frac{n}{2^i}$ 바이트보다 큰 경우, 데이터 메모리에 n 바이트의 기억영역을 반환하고, 프리 리스트 메모리에 의해 관리되는 n 바이트 엔트리 목록에 상기 기억영역에 대응되는 엔트리를 포함시키는 단계; 및 (b) 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 같거나 작은 경우, 데이터 메모리에 $\frac{n}{2^i}$ 바이트의 기억영역을 반환하고, 상기 프리 리스트 메모리에 의해 관리되는 $\frac{n}{2^i}$ 바이트 엔트리 목록에 상기 기억영역에 대응되는 엔트리를 포함시키되, 상기 반환된 기억영역과 동일한 엔트리로 관리되는 인접 기억 영역이 사용중이 아니면, 상기 반환된 기억영역 및 상기 인접 기억영역을 통합한 기억영역에 대응되는 엔트리를 상기 n 바이트 엔트리 목록에 포함시키는 단계를 포함하는 것을 특징으로 한다.

<18> 이하에서, 첨부된 도면을 참조하여 본 발명의 바람직한 실시예에 대하여 상세히 설명한다.

<19> 본 발명은 컴퓨터의 특정 응용 프로그램과 통신기기 등에서 자주 발생하는 일정한 범위에서 가변적인 크기로 이루어진 기억소자 할당 요구에 적합한 기억소자 할당 방법과

기구로 구성된다. 특히 데이터를 고정된 크기로 쪼개어 보관하는 경우, 하나의 데이터는 인덱스(index) 정보를 담는 헤더(header) 부분과 실제 쪼개어진 데이터가 담기는 페이로드(payload) 부분으로 나뉘어진 세그먼트(segment)들로 구성된다. 이 경우 각 세그먼트는 일정한 범위의 크기를 갖는다. 이러한 세그먼트의 기억소자로의 저장 요구는, 최대 크기가 정해진 상태에서 그 이하의 크기에 대한 할당을 요구하게 된다. 이와 같은 기억소자 할당에 대한 위치 주소값을 관리하는데 있어서, 종래에는 1 개의 고정 크기로 할당을 하던지, 아니면 소프트웨어적으로 관리하는 복잡한 방식으로 할당을 하여 왔다. 그러나, 본 발명에 따른 메모리 관리 장치 및 방법은, 메모리 할당에 대한 위치 주소 값을 하드웨어로 구현 가능하도록 복수 개의 고정된 크기로 나누어 관리하고, 가변 길이 패킷 저장을 위한 기억영역 할당시 기억영역 할당 요구에 가장 적합한 블록 혹은 서브 블록을 할당한다. 따라서, 고속의 기억영역 할당에 적합하면서도 기억영역을 효과적으로 사용할 수 있으며, 메모리의 낭비를 최소화할 수 있게 된다.

<20> 도 1은 데이터 메모리 블록 당 서브 블록이 4 개로 구성된, 본 발명의 바람직한 실시예에 따른 메모리 장치의 기억영역의 할당 예를 보여주는 블록도이다. 도 1을 참조하면, 본 발명에 따른 메모리 장치는, 프리 리스트 메모리(free lest memory ; 1100), 데이터 메모리(data memory ; 1200), 복수 개의 레지스터(1300), 및 주소변환기(1400)를 포함한다.

<21> 데이터 메모리(1200)는 실제 데이터들이 저장될 기억영역으로서, 소정의 길이를 가지는 복수 개의 서브 블록들로 구성된 복수 개의 블록들을 포함한다. 데이터 메모리(1200)는 가변 길이의 기억영역 할당 요구시 상기 서브 블록 및 상기 블록 중 어느 하나의 단위로 기억영역이 할당된다. 프리 리스트 메모리(1100)는 데이터 메모리(1200)를 관

리하기 위해 사용되는 메모리로써, 포인터를 활용하여 데이터 메모리(1200)의 빈 기억영역(즉, 사용이 종료된 기억영역)을 적어도 하나 이상의 목록(list)으로 관리한다. 레지스터(1300)는 프리 리스트 메모리(1100)에 저장된 목록의 시작점과 끝점을 저장하는데 사용되고, 주소변환기(1400)는 데이터 메모리(1200)와 프리 리스트 메모리(1100)간에 주소 값을 변환시키는데 사용된다. 이와 같은 구성을 가지는 메모리들(1100, 1200)과 레지스터(1300)는, 하나의 메모리 내에 각기 다른 주소를 부여받아 함께 존재할 수도 있고, 각기 다른 메모리로 구현될 수도 있다.

<22> 프리 리스트 메모리(1100)의 각 엔트리(entry) 수는 데이터 메모리(1200)의 엔트리 수와 일치하며, 상호간에 1:1 대응관계를 갖는다. 따라서, 프리 리스트 메모리(1100)의 엔트리 번호 1번에서 다루는 정보는 이 정보가 데이터 메모리(1200)의 엔트리 번호 1번에 해당되는 정보임을 의미한다. 그러므로, 이들간의 대응 관계를 위한 포인터는 따로 존재할 필요가 없게 되어, 기억영역을 절약할 수 있게 된다.

<23> 프리 리스트 메모리(1100)와 데이터 메모리(1200)와의 주소 대응관계를 살펴보면, 예를 들어, 데이터 메모리(1200)의 시작 주소가 0인 경우, 데이터 메모리(1200)를 구성하는 각 엔트리의 시작 주소는 "엔트리 번호 ×블록크기(n)"과 같은 간단한 규칙에 의해 계산된다. 데이터 메모리(1200)의 블록크기는 어떤 크기든 가능하지만, 보통 2의 제곱승 값을 취하므로, 주소변환기(140)는 이진수의 엔트리 번호에 0을 몇 개 붙이기만 하면 데이터 메모리(1200)와 프리 리스트 메모리(1100)간의 주소 값이 계산된다. 특히, 이를 하드웨어로 구현하는 경우, 하드웨어 상에서는 단순히 엔트리 번호에 해당되는 각 신호선을 상위 비트에 연결만 해 주면 구현이 가능하므로, 주소변환기 없이도 구현이 가능하다. 그리고 데이터 메모리(1200)의 시작 주소가 0이 아닌 경우, 데이터 메모리

(1200)의 시작 주소 값을 주소변환기(1400)에서 더해주기만 하면 된다. 즉, 이 경우의 데이터 메모리(1200)를 구성하는 각 엔트리의 시작 주소는 "엔트리 번호 x블록크기(n) + 데이터 메모리의 시작 주소 값"이 된다.

<24> 프리 리스트 메모리(1100)는 각각의 엔트리(1101, 1102, …) 별로 비트 마스크(bit mask)와 포인터 값들로 구성된다. 비트 마스크는 데이터 메모리(1200)의 각 블록에 포함된 각각의 서브 블록에 대응되어, 해당 서브 블록이 현재 사용 중인지 여부를 나타낸다.

<25> 도 1에서는 데이터 메모리(1200)의 블록(1201, 1202, …) 당 4 개의 서브 블록이 포함된 경우, 프리 리스트 메모리(1100)를 구성하는 각각의 엔트리(1101, 1102, …)가 4 비트의 비트 마스크를 갖는 예를 보여주고 있다. 데이터 메모리(1200)의 블록(1201, 1202, …)을 구성하는 서브 블록의 개수는 어떤 자연수 값이나 가능하지만, 가변 길이의 패킷을 저장하는 데 본 발명을 효과적으로 사용하기 위해서는 2, 4 ,8 같은 2의 제곱승 값이 유용하다.

<26> 프리 리스트 메모리(1100)를 구성하는 각각의 엔트리(1101, 1102, …)에는 다음의 빈 기억영역을 가리키는 포인터(NEXT_FREE_LIST_PTR) 값이 기본적으로 사용된다. 프리 리스트 메모리(1100)는 이 포인터를 활용하여 데이터 메모리(1200)의 빈 기억영역을 하나의 목록으로 유지하고, 사용자의 할당요구가 있을 때 목록의 맨 처음(header)부터 할당을 실시하게 된다.

<27> 레지스터(1300)는 이러한 할당요구에 대응하기 위해 목록의 처음 위치에 대한 포인터(Header Pointer of Free List ; HPFL)와, 목록의 끝 위치에 대한 포인터(Tail

Pointer of Free List ; TPFL)를 각각 보관한다. 예를 들어, 블록 당 서브 블록의 개수를 X라고 할 때, 레지스터(1300)는 "$1+\log_2 X$" 개의 포인터 쌍을 갖는다. 즉, 도 1과 같이 블록 당 서브 블록의 개수가 4 개인 경우, 상기 레지스터(1300)는 $1+\log_2 4 = 3$ 개의 포인터 쌍을 갖게 된다. 따라서, 참조번호 1301 내지 1306과 같이 6개의 포인터를 갖게 된다. 이 때, 제 1 및 제 2 포인터(1301, 1302)는 데이터 메모리(1200)에 대해 n 바이트 크기의 기억영역 할당에 관여한다. 그리고, 제 3 및 제 4 포인터(1303, 1304)는 n/2 바이트 크기의 기억영역 할당에 사용되고, 제 5 및 제 6 포인터(1305, 1306은) n/4 바이트 크기의 기억영역 할당에 각각 사용된다.

<28> 예를 들어, 블록의 크기가 256 바이트 (n=256)일 때, 129 내지 256 바이트 크기의 기억영역 할당 요구시에는 제 1 포인터(1HPFL ; 1301)의 값에 해당되는 엔트리를 할당(allocation)해 주고, 제 1 포인터(1301)의 값은 NEXT_FREE_LIST_PTR 값으로 갱신된다. 또한, 할당되었던 256 바이트의 블록이 반환(de-allocation)될 때는 제 2 포인터(1TPFL ; 1302)를 통해 프리 리스트 메모리(1100)에 추가된다. 같은 방식으로, 65 내지 128 바이트 크기의 기억영역의 할당은 제 3 포인터(2HPFL ; 1303)를 통해 수행되고, 할당되었던 128 바이트의 블록의 반환은 제 4 포인터(2TPFL ; 1304)를 통해 수행된다. 그리고, 64바이트 이하의 기억영역의 할당은 제 5 포인터(3HPFL ; 1305)를 통해 수행되고, 할당되었던 64 바이트의 블록의 반환은 제 6 포인터(3TPFL ; 1306)를 통해 수행된다.

<29> 이 때, 비트 마스크는 해당 데이터 메모리 블록이 사용될 때마다, 서브 블록 별로 해당 서브 블록이 현재 할당되어 사용되고 있는지 여부를 표시한다. 따라서, 할당되었던 기억영역이 반환될 때는 64 바이트가 반환이 되더라도, 동일 엔트리 내에 연속하는 서브 블록이 사용되지 않고 있으면(즉, 연속된 128 바이트의 기억영역이 사용되고 있지 않으

면), 해당 서브 블록은 64 바이트 영역으로 반환되는 대신, 사용되고 있지 않은 인접 64 바이트 영역과 함께 128 바이트 영역으로 반환되어, 데이터 메모리(1200)의 기억영역이 효과적으로 사용될 수 있게 된다. 이 때의 반환 동작에는, 64 바이트의 반환을 관리하는 제 6 포인터(1306) 대신 128 바이트의 반환을 관리하는 제 4 포인터(1304)가 사용된다. 이와 같은 본 발명에 따른 기억영역의 반환 규칙은, 64 바이트 영역과 128 바이트 영역 사이의 기억영역 반환은 물론, 128 바이트와 256 바이트 영역 사이의 기억영역 반환에도 마찬가지로 적용될 수 있다.

<30> 앞에서 설명한 바와 같이, 본 발명에 따른 메모리 관리 장치에 의하면, 데이터 메모리(1200)는, 기억영역 할당 요구시에는 가장 알맞은 크기의 블록 혹은 서브 블록으로 나뉘어 할당되고, 할당되었던 기억영역의 반환시에는 동일 엔트리 내에 연속하는 서브 블록의 사용여부에 따라 할당 가능한 최대 크기의 기억영역으로 반환된다. 따라서, 작은 크기의 기억영역의 할당 요구에 대해 효과적으로 대응할 수 있게 되고, 최적의 기억영역 활용이 가능해질 수 있게 된다.

<31> 도 2는 데이터 메모리의 블록 당 서브 블록이 2 개로 구성된, 본 발명의 바람직한 실시예에 따른 메모리 장치의 기억영역의 할당 예를 보여주는 블록도이다.

<32> 도 2를 참조하면, 데이터 메모리(2200)는 블록(2201, 2202, …) 당 2 개의 서브 블록들로 구성되며, 이에 대응되는 프리 리스트 메모리(2100)를 구성하는 각각의 엔트리(2101, 2102, …)는 2 비트의 비트 마스크를 포함한다. 이 경우, 데이터 메모리(2200)의 블록 당 서브 블록의 개수가 2이므로, 레지스터(2300)는 2개의 포인터 쌍을 포함한다. 여기서, 레지스터(2300)에 저장된 제 1 포인터(1HPFL ; 2301) 및 제 2 포인터(1TPFL ;

2302)는 128 바이트의 메모리 할당 및 반환에 관여하고, 제 3 포인터(2HPFL ; 2303) 및 제 4 포인터(2TPFL ; 2304)는 64 바이트의 메모리 할당 및 반환에 각각 관여한다.

<33> 도 2에는 데이터 메모리(2200)가 실제 사용 중일 때 프리 리스트 메모리(2100)의 구성 예가 도시되어 있다. 도 2에서, 프리 리스트 메모리(2100)의 비트 마스크 값이 "00"일 경우, 비트 마스크 값 "00"에 해당되는 기억영역(즉, 엔트리 번호 0x0000, 0x0001, 및 0x0003에 해당되는 기억 영역)은 128 바이트로 할당이 가능하므로, 프리 리스트 메모리(2100)는 도 2에 도시된 바와 같이 128 바이트의 기억영역에 대한 연결 목록(linked list)을 구성할 수 있다. 이 경우, 제 1 포인터(2301)는 목록의 맨 앞에 해당되는 0의 값(즉, 엔트리 번호 0x0000)을 갖게 된다.

<34> 그리고, 2 비트의 비트 마스크 중 한 비트가 1이고 나머지가 0인(즉, 비트 마스크가 "10" 또는 "01"인 경우) 엔트리에 해당되는 기억 영역(즉, 엔트리 번호 0x0002, 0x1FFE, 및 0x1FFF에 해당되는 기억 영역)은 64 바이트로 할당 가능하므로, 프리 리스트 메모리(2100)는 도 2에 도시된 바와 같이 64 바이트의 기억영역에 대한 연결 목록을 구성할 수 있다. 이 경우, 제 3 포인터(2303)는 해당 목록의 맨 앞의 값인 2의 값(즉, 엔트리 번호 0x0002)을 갖게 되고, 제 4 포인터(2304)는 해당 목록의 맨 마지막 값인 8091의 값(즉, 엔트리 번호 0x1FFF)을 갖게 된다. 이 때, 프리 리스트 메모리(2100)의 이전 포인터 값(PREV_PTR)은 꼭 필요한 것은 아니지만, 이전 포인터 값(PREV_PTR)이 존재할 경우, 엔트리를 목록에서 삭제할 때 해당 엔트리를 O(1)의 속도로 삭제하고 목록을 재구성할 수 있는 이점이 있다. 여기서, O(n)은 컴퓨터 알고리즘에서 수행속도를 나타내는 방식으로, n의 개수에 비례한 연산속도를 나타낸다. 그리고, $O(n^2)$는 n의 제곱승에 비례한 속도를 나타내며,

O(1)은 엔트리 개수에 관계없이 일정한 상수 값의 시간이 소요되는 것을 각각 나타낸다. 예를 들어, 이전 포인터 값(PREV_PTR)이 없을 때 목록에서 엔트리를 삭제하는 경우, 목록의 처음부터 그 위치를 찾아 목록을 재구성해야 하기 때문에 O(n)이 소요된다(여기서, n은 포인터로 연결된 목록상의 엔트리 개수). 이에 비해 이전 포인터 값(PREV_PTR)이 있는 경우, 엔트리의 삭제시 목록상의 위치를 찾는 노력이 필요치 않게 되어 곧바로 목록을 재구성할 수 있으므로, O(1)이 가능하게 되는 것이다.

<35> 도 3은 도 2에 도시된 메모리 장치를 위한 기억영역 할당 방법을 보여주기 위한 흐름도이다.

<36> 도 3을 참조하면, 기억영역의 할당을 위해서, 기억영역에 대한 할당 요구 크기가 64 바이트 이하인지 여부가 먼저 판별된다(3001 단계). 3001 단계에서의 판별 결과, 할당 요구 크기가 64 바이트 이하인 경우, 레지스터(2300)의 제 3 포인터(2HPFL ; 2303)에 해당되는 데이터 메모리(2200)의 해당 엔트리가 유효한지 여부(즉, 프리 리스트 메모리(2100)에 의해 관리되는 64 바이트 목록에 빈 기억영역이 있는지 여부)가 판별된다(3002 단계).

<37> 3002 단계에서의 판별 결과, 레지스터(2300)의 제 3 포인터(2HPFL)에 해당되는 데이터 메모리(2200)의 해당 엔트리가 유효한 경우, 제 3 포인터(2HPFL)에 해당되는 64 바이트의 엔트리를 기억영역으로 할당하고, 상기 엔트리를 구성하는 해당 서브 블록의 비트 마스크를 1로 설정한다(3203 단계). 그리고 나서 해당 엔트리의 NEXT_PTR 값으로 제 3 포인터(2HPFL)를 갱신함으로써, 64 바이트의 기억영역을 할당하게 된다(3204 단계).

<38> 그리고, 3002 단계에서의 판별 결과, 레지스터(2300)의 제 3 포인터(2HPFL)에 해당되는 데이터 메모리(2200)의 해당 엔트리가 유효하지 않은 경우, 즉, 프리 리스트 메모

리(2100)에 의해 관리되는 64 바이트 목록에 빈 기억영역이 없는 경우, 제 1 포인터(1HPFL)에 해당되는 128 바이트의 엔트리를 기억영역으로 할당하고, 상기 엔트리를 구성하는 해당 서브 블록의 비트 마스크를 1로 설정한다(3003 단계). 그리고, 할당된 엔트리를 64 바이트 목록을 관리하는 데 사용되는 제 4 포인터(2TPFL)에 추가함으로써, 상기 엔트리를 64 바이트 목록에 추가한다(3004 단계), 그리고 나서 해당 엔트리의 NEXT_PTR 값으로 제 1 포인터(1HPFL)를 갱신함으로써, 128바이트 목록에서 해당 엔트리를 삭제시킨다(3005 단계). 그 결과, 128 바이트의 반은 기억영역으로 할당되고, 나머지 반은 다음에 64 바이트의 할당에 사용될 수 있게 된다.

<39> 그리고, 3001 단계에서의 판별 결과, 기억 용량의 할당이 요구된 크기가 64 바이트 보다 크면, 레지스터(2300)의 제 1 포인터(즉, 1HPFL ; 2301)에 해당되는 128 바이트의 엔트리를 할당하고, 상기 엔트리에 대응되는 비트 마스크를 1로 설정한다(3102 단계). 그리고 나서 해당 엔트리의 NEXT_PTR 값으로 제 1 포인터(1HPFL)를 갱신함으로써, 128 바이트의 기억영역을 할당하게 된다(3103 단계).

<40> 앞에서 설명한 바와 같이, 본 발명에 따른 기억영역 할당 방법은, 기억영역의 할당 요구 크기에 따라 각기 다른 크기를 가지는 소정의 기억영역이 할당된다. 예를 들어, 기억영역의 할당 요구 크기가 64 바이트보다 크면 128 바이트의 기억영역 할당을 실시하고, 기억영역의 할당 요구 크기가 64 바이트 이하이면 64 바이트의 기억영역 할당을 실시한다. 이 때, 할당되는 기억영역의 블록 사이즈는 사용자에 의해 더 세분화될 수 있다. 특히, 본 발명에 따른 기억영역의 할당 방법에서는 64 바이트 이하의 기억영역 할당시, 64 바이트 목록에 유효 엔트리가 존재하지 않는 경우, 128 바이트의 목록을 분할하여 64 바이트의 기억영역으로 할당한다. 따라서, 본 발명에 따른 기억영역 할당 방

법은, 기본적으로는 128 바이트의 관리에 필요한 관리 정보의 분량만을 가지고도 64 바이트 및 128 바이트 모두에 관한 기억영역 할당을 수행할 수 있다.

<41> 도 4는 도 2에 도시된 메모리 장치를 위한 기억영역 반환 방법을 보여주기 위한 흐름도로서, 데이터 메모리(2200)의 블록 당 2 개의 서브 블록이 포함된 경우의 기억영역 반환 절차를 보여주고 있다.

<42> 도 4를 참조하면, 기억영역의 반환을 위해서, 반환되는 기억영역의 크기가 64 바이트 이하인지 여부가 먼저 판별된다(4001 단계). 4001 단계에서의 판별 결과, 반환되는 기억영역의 크기가 64 바이트 이하이면, 반환되는 엔트리의 비트 마스크가 모두 1로 설정되어 있는지 여부(즉, 해당 엔트리가 128 바이트의 기억영역으로 사용될 수 있도록 반환될 수 있는지의 여부)가 판별된다(4002 단계).

<43> 4002 단계에서의 판별 결과, 반환되는 엔트리의 비트 마스크가 모두 1로 설정되어 있는 경우(즉, 해당 엔트리가 64 바이트의 기억영역으로 반환되는 경우), 반환되는 엔트리의 해당 비트 마스크를 0으로 설정하고, NEXT_PTR을 -1로 설정한다(4203 단계). 그리고 나서 반환되는 엔트리를 제 4 포인터(2TPFL ; 2304)에 해당되는 엔트리의 NEXT_PTR로 설정함으로써, 해당 엔트리를 64 바이트 목록에 추가시킨다(4204 단계).

<44> 그리고, 4002 단계에서의 판별 결과, 반환되는 엔트리의 비트 마스크가 모두 1로 설정되어 있지 않은 경우(즉, 해당 엔트리가 128 바이트의 기억영역으로 반환될 수 있는 경우), 반환되는 엔트리를 64 바이트 목록에서 삭제한다(4003 단계). 그리고, 반환되는 엔트리의 해당 비트 마스크를 0으로 설정하고, NEXT_PTR을 -1로 설정한다(4004 단계). 그리고 나서, 반환되는 엔트리를 제 2 포인터(1TPFL ; 2302)에 해당되는 엔트리의 NEXT_PTR로 설정함으로써, 해당 엔트리를 128 바이트 목록에 추가시킨다(4005 단계).

<45> 그리고, 4001 단계에서의 판별 결과, 반환되는 기억영역의 크기가 64 바이트 보다 크면, 반환되는 엔트리의 해당 비트 마스크를 0으로 설정하고, NEXT_PTR을 -1로 설정한다(4102 단계). 그리고 나서, 반환되는 엔트리를 제 2 포인터(1TPFL ; 2302)에 해당되는 엔트리의 NEXT_PTR로 설정함으로써, 해당 엔트리를 128 바이트 목록에 추가시킨다(4103 단계).

<46> 앞에서 설명한 바와 같이, 본 발명에 따른 기억영역 반환 방법은, 반환되는 기억영역의 크기에 따라서 각기 다른 크기를 가지는 소정의 기억영역이 반환된다. 예를 들어, 기억영역의 반환 요구 크기가 64 바이트보다 크면 128 바이트의 기억영역이 반환되고, 기억영역의 할당 요구 크기가 64 바이트 이하이면 64 바이트의 기억영역이 반환된다. 본 발명에서는, 특히 64 바이트 이하의 기억영역의 반환시, 반환되는 64 바이트의 동일 엔트리 내에 인접해 있는 64 바이트의 기억영역이 사용중이 아니면(즉, 비어 있는 상태이면), 상기 반환되는 기억영역과, 상기 인접 기억영역을 두 개의 64 바이트 기억영역으로 반환하여 사용하는 대신, 128 바이트의 기억영역 하나로 반환하여 사용한다. 따라서, 본 발명에 따른 기억영역 할당 방법은, 항상 할당 가능한 최대 크기의 기억영역으로 반환될 수 있게 되어, 최적의 기억영역 활용이 가능해진다.

<47> 이와 같은 기억영역의 할당 및 반환 방법은, 현재 네트워크 장비 내에서 인터넷의 데이터 전달의 대부분을 차지하고 있는 IP 패킷(Internet Protocol Packet)의 저장에 알맞은 구조이다. IP패킷의 크기는 이론적으로는 20 바이트(즉, IP 패킷의 헤더 크기)에서 64 킬로바이트까지 가능하나, 실제로는 평균 300 바이트의 크기의 패킷이 주로 사용되고 있으며, 40 바이트 크기를 갖는 패킷이 특히 많은 분포를 갖고 있다.

<48> 따라서, 현재 대부분의 네트워크 장비에서는 128 바이트나 256 바이트의 고정된 세그먼트 단위로 패킷을 쪼개어 저장하고 있다. 따라서, 패킷의 맨 마지막 세그먼트의 대부분은 낭비되고 있으며, 40 바이트의 크기를 갖는 패킷이 특히 많은 분포하는 IP 패킷의 특성을 고려할 때, 256 바이트나 128 바이트의 세그먼트 단위로 패킷을 나누게 되면, 실제적으로 반 이상의 기억영역을 낭비하게 된다.

<49> 따라서, 본 발명을 네트워크 장비에 적용하게 되면, 도 2에서 알 수 있는 바와 같이, 128 바이트의 세그먼트 관리 정보만으로도 64바이트 단위의 기억영역 할당이 가능하게 되어, 40 바이트 패킷에서 초래되는 기억영역의 낭비를 크게 줄일 수 있다. 특히, 본 발명은 고속으로 패킷 저장을 수행해야 하는 네트워크 장비를 위해 하드웨어적인 로직으로 구현이 가능하므로, 고속의 기억영역 할당에 적합하면서도 기억영역을 효과적으로 사용할 수 있는 장점이 있다.

<50> 이상에서, 본 발명의 실시예로서 네트워크 장비에서의 IP 패킷에 대한 기억영역 할당 및 반환에 대해 구체적으로 예시되었으나, 그밖에도 기억소자(즉, 메모리)를 활용하는 다양한 기기에서 수행될 수 있는 다양한 길이의 기억영역의 할당 및 반환에도 본 발명을 적용할 수 있다.

<51> 본 발명은 또한 컴퓨터로 읽을 수 있는 기록매체에 컴퓨터가 읽을 수 있는 코드로서 구현하는 것이 가능하다. 컴퓨터가 읽을 수 있는 기록매체는 컴퓨터 시스템에 의하여 읽혀질 수 있는 데이터가 저장되는 모든 종류의 기록장치를 포함한다. 컴퓨터가 읽을 수 있는 기록매체의 예로는 ROM, RAM, CD-ROM, 자기 테이프, 플로피디스크, 광데이터 저장장치 등이 있으며, 또한 캐리어 웨이브(예를 들어 인터넷을 통한 전송)의 형태로 구현되는 것도 포함한다. 또한 컴퓨터가 읽을 수 있는 기록매체는 네트워크로 연결된 컴퓨터

시스템에 분산되어, 분산방식으로 컴퓨터가 읽을 수 있는 코드로 저장되고 실행될 수 있다.

【발명의 효과】

<52> 이상에 설명한 바와 같이, 본 발명에 의한 가변 길이의 패킷 저장을 위한 메모리 관리 장치 및 방법에 의하면, 메모리 할당에 대한 위치 주소 값은 하드웨어로 구현 가능하도록 복수 개의 고정된 크기로 나누어 관리된다. 따라서, 특정한 크기 이하의 가변 길이의 패킷 저장을 위한 기억영역 할당시에는 기억영역 할당 요구에 가장 적합한 블록 혹은 서브 블록이 할당되고, 할당되었던 기억영역의 반환시에는 동일 엔트리 내에 연속하는 서브 블록의 사용여부에 따라 할당 가능한 최대 크기의 기억영역으로 반환된다. 그러므로, 작은 크기의 기억영역의 할당 요구에 대해 효과적으로 대응할 수 있게 되고, 최적의 기억영역 활용이 가능해질 수 있게 되어, 메모리의 낭비를 최소화할 수 있게 된다.

【특허청구범위】

【청구항 1】

소정의 길이를 가지는 복수 개의 서브 데이터 블록들을 구비한 복수 개의 데이터 블록들을 구비하여, 가변 길이의 기억영역 할당 요구시 상기 서브 데이터 블록 및 상기 데이터 블록 중 어느 하나의 단위로 기억영역이 할당되는 데이터 메모리;

포인터를 이용하여 상기 데이터 메모리의 빈 기억영역을 적어도 하나 이상의 목록으로 관리하는 프리 리스트 메모리; 및

상기 목록의 시작 위치정보 및 끝 위치정보를 저장하기 위한 레지스터를 포함하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 2】

제 1 항에 있어서,

상기 메모리 관리 장치는 상기 데이터 메모리 및 상기 프리 리스트 메모리 사이의 주소 값 변환을 수행하는 주소변환기를 더 포함하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 3】

제 1 항에 있어서,

상기 프리 리스트 메모리의 엔트리 수는 상기 데이터 메모리의 엔트리 수와 일치하며, 상기 프리 리스트 메모리 및 상기 데이터 메모리에 포함된 상기 엔트리들은 각각 1:1 대응 관계를 갖는 것을 특징으로 하는 메모리 관리 장치.

【청구항 4】

제 3 항에 있어서,

상기 데이터 메모리를 구성하는 각 엔트리의 시작 주소는 "엔트리 번호 ×데이터 블록크기(n) + 데이터 메모리의 시작 주소 값"을 가지는 것을 특징으로 하는 메모리 관리 장치.

【청구항 5】

제 1 항에 있어서,

상기 데이터 메모리는, n이 2의 배수이고, i 및 j가 각각 양의 정수일 때(단, i< j), n 바이트의 기억영역을 가지는 복수 개의 데이터 블록들을 포함하며,

상기 각각의 데이터 블록은 $\frac{n}{2^i}$ 바이트의 기억영역을 가지는 복수 개의 서브 데이터 블록들로 구성되고, 상기 각각의 서브 데이터 블록들은 $\frac{n}{2^j}$ 바이트의 기억영역을 가지는 복수 개의 서브 데이터 블록들로 각각 구성되는 계층적 구조를 가지는 것을 특징으로 하는 메모리 관리 장치.

【청구항 6】

제 1 항에 있어서, 상기 프리 리스트 메모리를 구성하는 각각의 엔트리는

상기 각각의 서브 데이터 블록의 사용 여부를 나타나는 복수 개의 비트 마스크; 및

상기 목록에서 현재 선택된 엔트리 바로 뒤에 위치한 엔트리를 가리키는 제 1 포인터를 포함하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 7】

제 1 항에 있어서,

상기 프리 리스트 메모리를 구성하는 각각의 엔트리는, 상기 목록에서 현재 선택된 엔트리 바로 앞에 위치한 엔트리를 가리키는 제 2 포인터를 더 포함하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 8】

제 7 항에 있어서,

상기 프리 리스트 메모리는, 상기 비트 마스크 값에 따라 n 바이트의 기억영역이 할당 가능한 n 바이트 엔트리 목록, $\frac{n}{2^1}$ 바이트의 기억영역이 할당 가능한 $\frac{n}{2^1}$ 바이트 엔트리 목록, 및 $\frac{n}{2^1}$ 바이트의 기억영역이 할당 가능한 $\frac{n}{2^1}$ 바이트 엔트리 목록을 각각 구성하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 9】

제 1 항에 있어서,

상기 레지스터는, 상기 데이터 블록이 X 개의 서브 데이터 블록들로 구성되는 경우, 상기 목록의 상기 시작 위치정보 및 상기 끝 위치정보를 저장하기 위한 $1+\log_2 X$ 개의 포인터 쌍들을 포함하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 10】

제 8 항에 있어서,

상기 메모리 관리 장치는, 상기 $\frac{n}{2^1}$ 바이트의 기억영역 할당시 상기 $\frac{n}{2^1}$ 바이트

엔트리 목록에 유효 엔트리가 존재하지 않을 경우, 상기 n 바이트 엔트리 목록을 분할하여 상기 $\frac{n}{2^i}$ 바이트의 기억영역으로 할당하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 11】

제 8 항에 있어서,

상기 메모리 관리 장치는, 상기 $\frac{n}{2^j}$ 바이트의 기억영역 할당시 상기 $\frac{n}{2^j}$ 바이트 엔트리 목록에 유효 엔트리가 존재하지 않을 경우, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록을 분할하여 상기 $\frac{n}{2^j}$ 바이트의 기억영역으로 할당하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 12】

제 8 항에 있어서,

상기 메모리 관리 장치는, 상기 $\frac{n}{2^i}$ 바이트의 기억영역 반환시 상기 반환되는 기억영역의 동일 엔트리 내에 인접한 $\frac{n}{2^i}$ 바이트의 기억영역이 사용 중이 아니면, 상기 반환되는 기억영역 및 상기 인접 기억영역을 통합하여 상기 n 바이트의 기억영역으로 반환하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 13】

제 8 항에 있어서,

상기 메모리 관리 장치는, 상기 $\frac{n}{2^j}$ 바이트의 기억영역 반환시 상기 반환되는 기억영역의 동일 엔트리 내에 인접한 $\frac{n}{2^j}$ 바이트의 기억영역이 사용 중이 아니면, 상기 반환되는 기억영역 및 상기 인접 기억영역을 통합하여 상기 $\frac{n}{2^i}$ 바이트의 기억영역으로 반환하는 것을 특징으로 하는 메모리 관리 장치.

【청구항 14】

(a) n이 2의 배수이고 i가 양의 정수일 때, 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 큰 경우, 프리 리스트 메모리에 의해 관리되는 n 바이트 엔트리 목록 상에 존재하는 유효 엔트리에 n 바이트의 기억영역을 할당하는 단계; 및

(b) 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 같거나 작은 경우, 상기 프리 리스트 메모리에 의해 관리되는 $\frac{n}{2^i}$ 바이트 엔트리 목록 상에 존재하는 유효 엔트리에 $\frac{n}{2^i}$ 바이트의 기억영역을 할당하되, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록에 유효 엔트리가 존재하지 않으면, 상기 n 바이트 엔트리 목록을 분할하여 $\frac{n}{2^i}$ 바이트의 기억영역으로 할당하는 단계를 포함하는 것을 특징으로 하는 메모리 할당 방법.

【청구항 15】

제 14 항에 있어서, 상기 (a) 단계는

(a-1) 상기 n 바이트 엔트리 목록의 시작 위치에 해당되는 엔트리를 상기 기억영역으로 할당하고, 상기 기억영역에 대응되는 비트마스크를 상기 엔트리가 현재 사용중임을 나타내는 제 1의 값으로 설정하는 단계; 및

(a-2) 상기 n 바이트 엔트리 목록의 시작 위치 값을 상기 엔트리와 동일한 엔트리 목록에 속하는 다음 엔트리의 위치 값으로 갱신하는 단계를 포함하는 것을 특징으로 하는 메모리 할당 방법.

【청구항 16】

제 14 항에 있어서, 상기 (b) 단계는

(b-1) 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록에 유효 엔트리가 존재하는지 여부를 판별하는 단계;

(b-2) 상기 (b-1) 단계에서의 판별 결과, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록에 유효 엔트리가 존재하는 경우, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록의 시작 위치에 해당되는 엔트리를 상기 기억영역으로 할당하고, 상기 기억영역에 대응되는 비트마스크를 상기 제 1의 값으로 설정하는 단계;

(b-3) 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록의 시작 위치 값을 상기 엔트리와 동일한 엔트리 목록에 속하는 다음 엔트리의 위치 값으로 갱신하는 단계;

(b-4) 상기 (b-1) 단계에서의 판별 결과, 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록에 유효 엔트리가 존재하지 않는 경우, 상기 n 바이트 엔트리 목록의 시작 위치에 해당되는 엔트리를 상기 기억영역으로 할당하고, 상기 기억영역에 대응되는 비트마스크를 상기 제 1의 값으로 설정하는 단계;

(b-5) 할당된 상기 엔트리의 위치 값을 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록의 끝 위치에 추가시키는 단계; 및

(b-6) 상기 n 바이트 엔트리 목록의 시작 위치 값을 상기 엔트리와 동일한 엔트리 목록에 속하는 다음 엔트리의 위치 값으로 갱신하는 단계를 포함하는 것을 특징으로 하는 메모리 할당 방법.

【청구항 17】

(a) n이 2의 배수이고 i가 양의 정수일 때, 반환되는 기억영역의 크기가 $\frac{n}{2^i}$ 바이트보다 큰 경우, 데이터 메모리에 n 바이트의 기억영역을 반환하고, 프리 리스트 메모리에 의해 관리되는 n 바이트 엔트리 목록에 상기 기억영역에 대응되는 엔트리를 포함시키는 단계; 및

(b) 기억영역의 할당 요구 크기가 $\frac{n}{2^i}$ 바이트보다 같거나 작은 경우, 데이터 메모리에 $\frac{n}{2^i}$ 바이트의 기억영역을 반환하고, 상기 프리 리스트 메모리에 의해 관리되는 $\frac{n}{2^i}$ 바이트 엔트리 목록에 상기 기억영역에 대응되는 엔트리를 포함시키되, 상기 반환된 기억영역과 동일한 엔트리로 관리되는 인접 기억 영역이 사용중이 아니면, 상기 반환된 기억영역 및 상기 인접 기억영역을 통합한 기억영역에 대응되는 엔트리를 상기 n 바이트 엔트리 목록에 포함시키는 단계를 포함하는 것을 특징으로 하는 메모리 반환 방법.

【청구항 18】

제 17 항에 있어서, 상기 (a) 단계는

(a-1) 상기 반환되는 기억영역에 대응되는 데이터 블록의 비트마스크를 상기 엔트리가 현재 사용중이지 않음을 나타내는 제 2의 값으로 설정하고, 상기 기억영역 다음에 사용될 기억영역의 위치 값을 유효 엔트리가 없음을 나타내는 제 3의 값으로 설정하는 단계; 및

(a-2) 상기 반환되는 기억영역을 상기 n 바이트 엔트리 목록의 끝 위치에 추가시키는 단계를 포함하는 것을 특징으로 하는 메모리 반환 방법.

【청구항 19】

제 17 항에 있어서, 상기 (b) 단계는

(b-1) 상기 반환되는 기억영역이 속하는 엔트리에 포함된 $\frac{n}{2^i}$ 바이트의 서브 데이터 블록들이 모두 사용중인지 여부를 판별하는 단계;

(b-2) 상기 (b-1) 단계에서의 판별 결과, 상기 서브 데이터 블록들이 모두 사용중인 경우, 상기 반환되는 기억영역에 대응되는 상기 서브 데이터 블록의 비트마스크를 상기 제 2의 값으로 설정하고, 상기 기억영역 다음에 사용될 기억영역의 위치 값을 상기 제 3의 값으로 설정하는 단계;

(b-3) 상기 반환되는 기억영역을 상기 $\frac{n}{2^i}$ 바이트 엔트리 목록의 끝 위치에 추가시키는 단계;

(b-4) 상기 (b-1) 단계에서의 판별 결과, 상기 서브 데이터 블록들이 모두 사용중이 아닌 경우, 상기 반환되는 기억영역에 대응되는 엔트리를 상기 $\frac{n}{2^i}$ 바이트 목록에서 삭제하는 단계;

(b-5) 상기 반환되는 기억영역이 포함된 상기 데이터 블록의 비트마스크를 상기 제 2의 값으로 설정하고, 상기 기억영역 다음에 사용될 기억영역의 위치 값을 상기 제 3의 값으로 설정하는 단계; 및

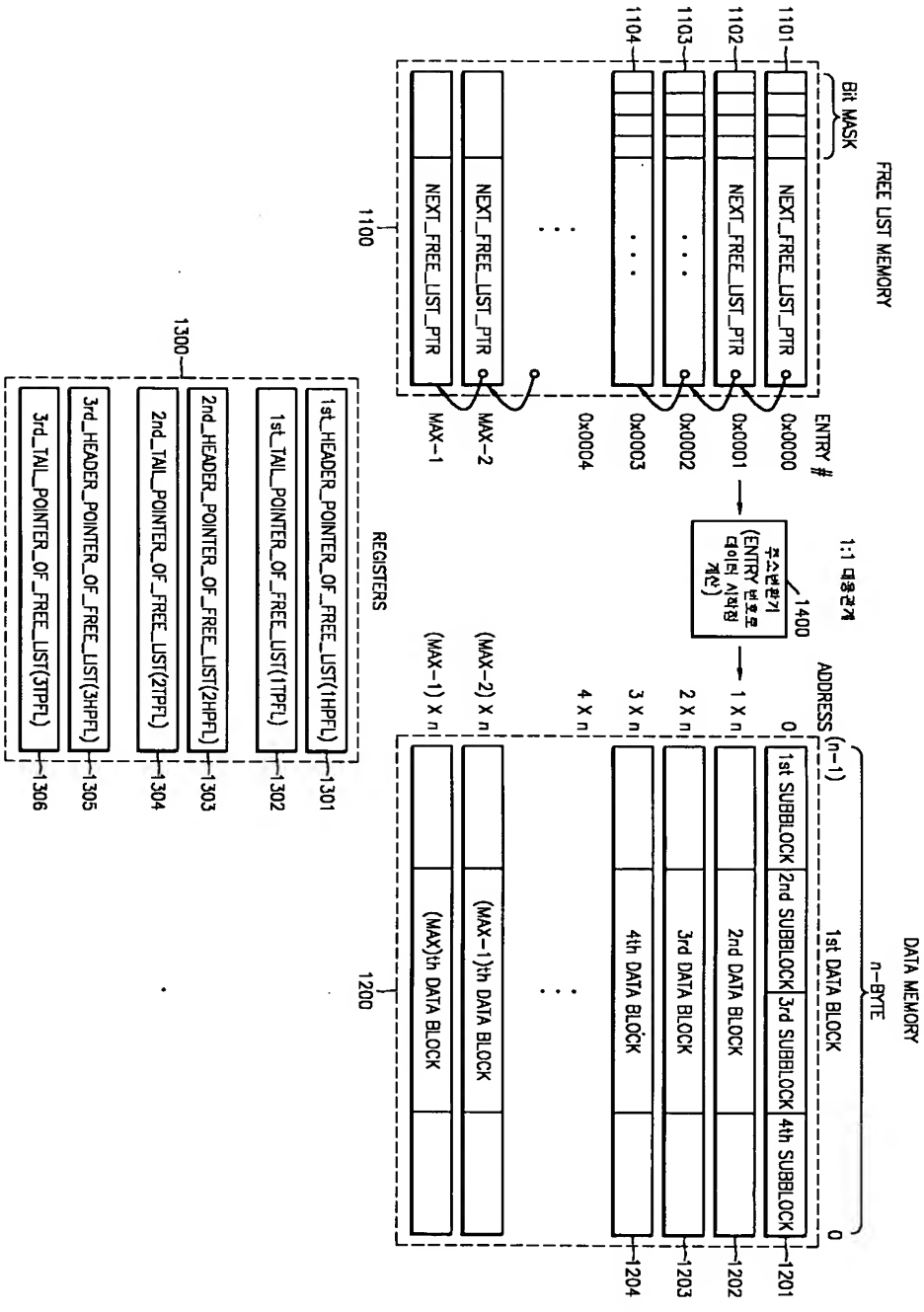
(b-6) 상기 반환되는 기억영역을 상기 n 바이트 엔트리 목록의 끝 위치에 추가시키는 단계를 포함하는 것을 특징으로 하는 메모리 반환 방법.

【청구항 20】

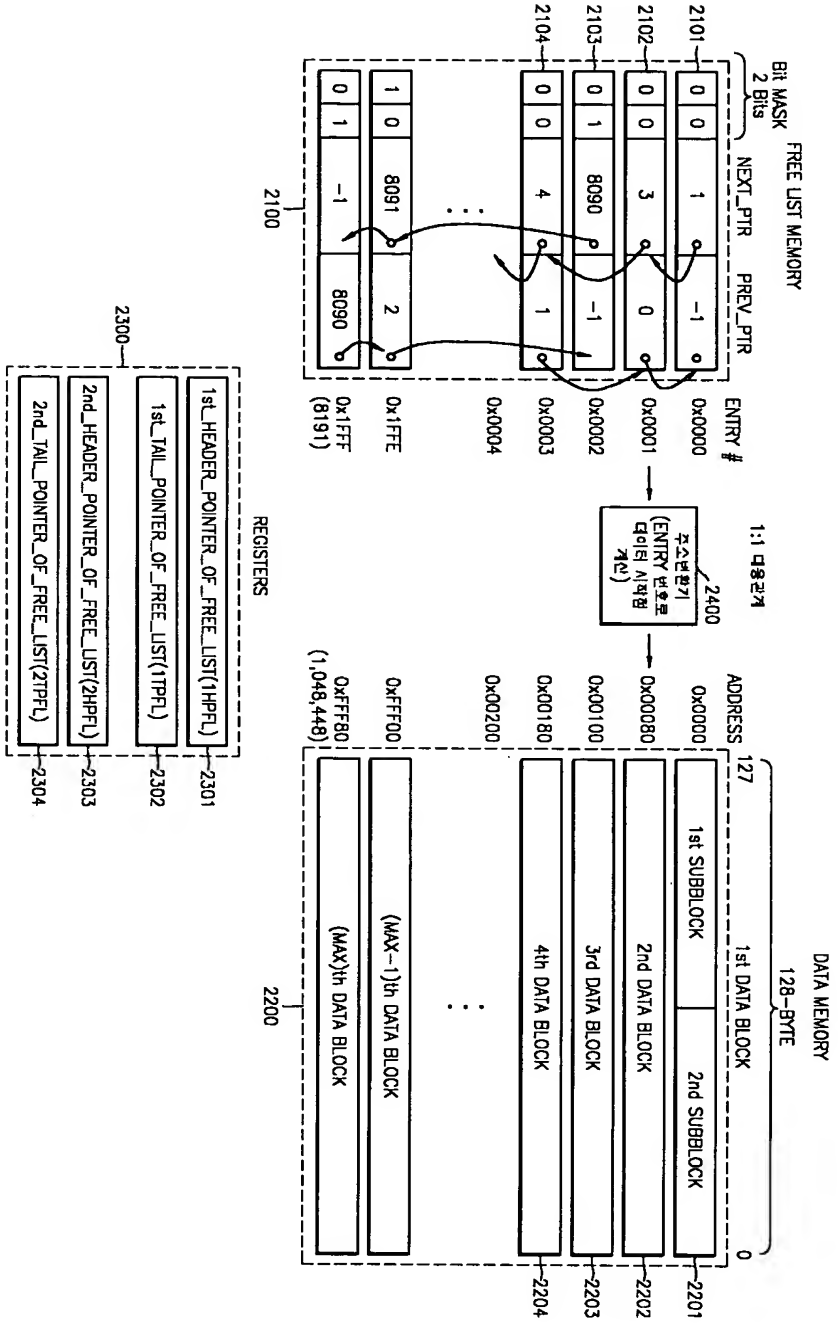
제 14 항 내지 제 19 항 중 어느 한 항의 방법을 컴퓨터에서 실행시키기 위한 프로그램을 기록한 컴퓨터로 읽을 수 있는 기록 매체.

【도면】

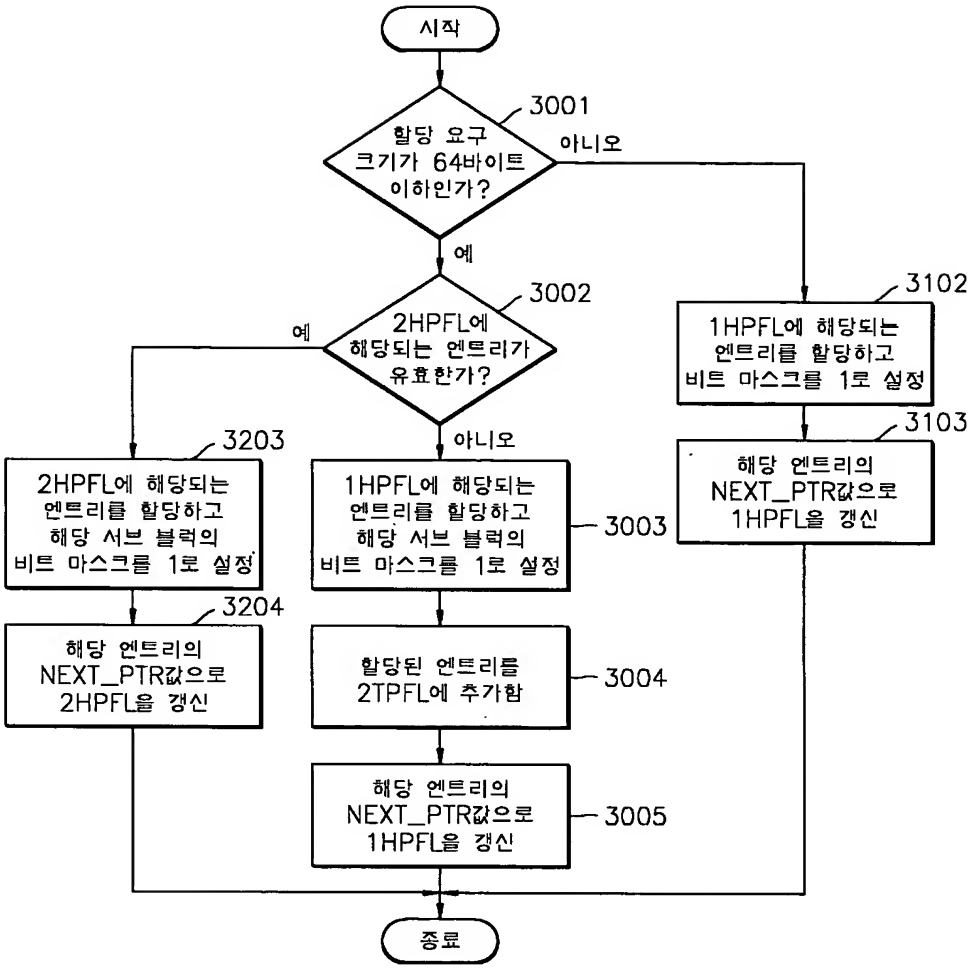
【도 1】


FREE LIST MEMORY
Bit MASK
NEXT_FREE_LIST_PTR
NEXT_FREE_LIST_PTR
NEXT_FREE_LIST_PTR
NEXT_FREE_LIST_PTR
1101
1102
1103
1104
1100
ENTRY #
0x0000
0x0001
0x0002
0x0003
0x0004
MAX-1
MAX-2
1:1 대응관계
주소변환기 (ENTRY 번호로 데이터 시작점 계산)
1400
REGISTERS
1st_HEADER_POINTER_OF_FREE_LIST(1HPFL)
1st_TAIL_POINTER_OF_FREE_LIST(1TPFL)
2nd_HEADER_POINTER_OF_FREE_LIST(2HPFL)
2nd_TAIL_POINTER_OF_FREE_LIST(2TPFL)
3rd_HEADER_POINTER_OF_FREE_LIST(3HPFL)
3rd_TAIL_POINTER_OF_FREE_LIST(3TPFL)
1300
1301
1302
1303
1304
1305
1306
1200
ADDRESS
0
1 X n
2 X n
3 X n
4 X n
(MAX-2) X n
(MAX-1) X n
(n-1)
DATA MEMORY
n-BYTE
1st SUBBLOCK
2nd SUBBLOCK
3rd SUBBLOCK
4th SUBBLOCK
1st DATA BLOCK
2nd DATA BLOCK
3rd DATA BLOCK
4th DATA BLOCK
(MAX-1)th DATA BLOCK
(MAX)th DATA BLOCK
1201
1202
1203
1204

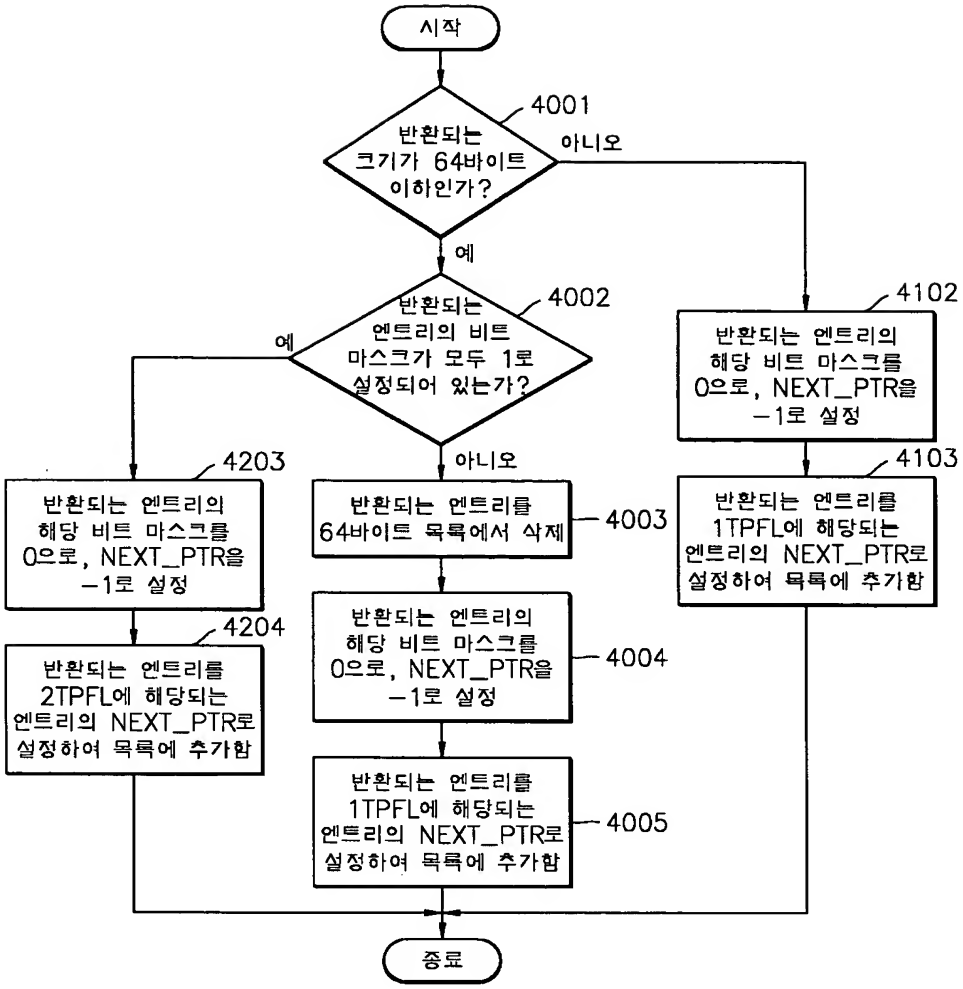
【도 2】


FREE LIST MEMORY
Bit MASK 2 Bits
NEXT_PTR
PREV_PTR
ENTRY #
2101
2102
2103
2104
2100
0x0000
0x0001
0x0002
0x0003
0x0004
0x1FFE
0x1FFF (8191)
1:1 대응관계
주소변환기 (ENTRY 번호로 데이터 시작점 계산)
2400
DATA MEMORY
128-BYTE
ADDRESS
1st DATA BLOCK
2nd SUBBLOCK
1st SUBBLOCK
2nd DATA BLOCK
3rd DATA BLOCK
4th DATA BLOCK
(MAX-1)th DATA BLOCK
(MAX)th DATA BLOCK
0x0000
0x00080
0x00100
0x00180
0x00200
0xFFF00
0xFFF80 (1,048,448)
2200
2201
2202
2203
2204
REGISTERS
2300
1st_HEADER_POINTER_OF_FREE_LIST(1HPFL)
1st_TAIL_POINTER_OF_FREE_LIST(1TPFL)
2nd_HEADER_POINTER_OF_FREE_LIST(2HPFL)
2nd_TAIL_POINTER_OF_FREE_LIST(2TPFL)
2301
2302
2303
2304

【도 3】


시작
3001
할당 요구 크기가 64바이트 이하인가?
아니오
예
3002
2HPFL에 해당되는 엔트리가 유효한가?
예
아니오
3102
1HPFL에 해당되는 엔트리를 할당하고 비트 마스크를 1로 설정
3103
해당 엔트리의 NEXT_PTR값으로 1HPFL을 갱신
3203
2HPFL에 해당되는 엔트리를 할당하고 해당 서브 블럭의 비트 마스크를 1로 설정
3204
해당 엔트리의 NEXT_PTR값으로 2HPFL을 갱신
1HPFL에 해당되는 엔트리를 할당하고 해당 서브 블럭의 비트 마스크를 1로 설정
3003
할당된 엔트리를 2TPFL에 추가함
3004
해당 엔트리의 NEXT_PTR값으로 1HPFL을 갱신
3005
종료

【도 4】


시작
4001
반환되는 크기가 64바이트 이하인가?
아니오
예
4002
반환되는 엔트리의 비트 마스크가 모두 1로 설정되어 있는가?
예
아니오
4102
반환되는 엔트리의 해당 비트 마스크를 0으로, NEXT_PTR을 -1로 설정
4103
반환되는 엔트리를 1TPFL에 해당되는 엔트리의 NEXT_PTR로 설정하여 목록에 추가함
4203
반환되는 엔트리의 해당 비트 마스크를 0으로, NEXT_PTR을 -1로 설정
4204
반환되는 엔트리를 2TPFL에 해당되는 엔트리의 NEXT_PTR로 설정하여 목록에 추가함
4003
반환되는 엔트리를 64바이트 목록에서 삭제
4004
반환되는 엔트리의 해당 비트 마스크를 0으로, NEXT_PTR을 -1로 설정
4005
반환되는 엔트리를 1TPFL에 해당되는 엔트리의 NEXT_PTR로 설정하여 목록에 추가함
종료